Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 4 (2011) 66–75

Procedia Computer Science

International Conference on Computational Science, ICCS 2011

Cellular Microscopic Pattern Recogniser – A Distributed Computational Approach for Macroscopic Event Detection in WSN

Waleed M. Alfehaid and Asad I. Khan

Clayton school of IT, Monash University, Victoria, 3800, Australia

Abstract

A light weight one-shot learning pattern recognition scheme, known as Cellular Microscopic Pattern Recogniser (CMPR) is proposed that will lead to macroscopic event detection with wireless sensor networks (WSN). The scheme simplifies computations for energy conservation and speeds up recognition by leveraging the parallel distributed processing capabilities of WSN. Experimental results show that the scheme is able to detect noisy patterns with comparable accuracy to the conventional techniques.

"Keywords: Distributed Pattern Recognition; Evolutionary Computing; Artificial Neural Network; Wireless Sensor Network;"

1. Introduction

Event detection applications are most likely to be real-time where the network is analysing the sensory data to detect a specific or a set of related events. For example, in the field of structural health monitoring (SHM) Kim, et al. [1] deployed WSN on the Golden Gate Bridge to obtain and analyse ambient vibrations of the bridge. In the same field, Kijewski-Correa, et al. [2] presented a multi-scale WSN approach to improve damage detection in SHM. Another example is the work of Lymberopoulos, et al. [3]. They deployed sensor networks inside homes to monitor elderly people and detect their normal and abnormal activities. These applications are expected to be mission critical and require accurate and quick recognition and reporting in noisy environments [4], [5].

Event detection can be performed by recognising patterns associated with specific events. Pattern recognition in WSN is highly sensitive to computation complexity, number of iterations, and nodes training requirements owing to the resource constraints within WSN. Conventional computer based machine learning techniques are tightly coupled and iterative. Hence, these techniques do not suit the finely distributed architecture of WSN.

This paper will introduce a novel computational scheme, known as Cellular Microscopic Pattern Recogniser (CMPR) that addresses the real time mission critical applications requirements in resource constraints WSN. The proposed scheme distributes the computations among a network's nodes and allows loosely coupled communication which would lead to global and fast recognition with reducing computational constraints. The paper is organised as

* Corresponding author: Waleed M. Alfehaid. Tel: +61399055778; fax: +61399055159.
E-mail address: waleed.alfehaid@monash.edu.

1877–0509 © 2011 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
Selection and/or peer-review under responsibility of Prof. Mitsuhisa Sato and Prof. Satoshi Matsuoka
doi:10.1016/j.procs.2011.04.008

follows. Section 2 presents the existing work related to pattern recognition in WSN. Section 3 presents the proposed scheme, its architecture and communication stages. In Section 4, we test the CMPR as a competitive pattern recogniser. Section 5 concludes the paper.

2. Pattern recognition schemes for WSN

Existing pattern recognition techniques for WSNs include threshold-based, statistical, syntactical, associative memory, and graph neuron. Threshold-based are considered to be the simplest and widely used pattern recognition techniques in WSNs. Each sensor in such techniques is assigned a threshold value or in some cases more than one threshold values. When a sensor's reading hits the threshold value, it declares the recognition of the pattern of interest. For instance, Kim, et al. [6] presented a fence surveillance model that can detect intruders based on thresholds obtained from the average signal measurements of each sensor. If a node's reading exceeds its threshold, it sends a DETECT signal to the base station. Threshold-based techniques are considered to be light-weight and simple. However, these techniques are limited in terms of dealing with noisy patterns and thus, could lead to false alarms [7].

Statistical pattern recognition schemes are dependent on the probability of occurrence of a pattern. The theory of this approach is based on two major assumptions: recognition decision is achieved in terms of probability, and the probabilities of occurrence values are known [8]. Using such model requires pre-knowledge and historical study of the patterns in order to determine priority and conditional probability values. Additionally, it will require vast and centralised computational resources.

The Syntactic model describes the relationship between sub-patterns and patterns by creating structural rules. It adopts the language theory where letters form words and words form sentences based on grammatical rules. In this model, primitive elements and sub-patterns relationships are analysed to provide pattern recognition. The main constraints in the syntactic approach in describing the relationships (rules) between sub-patterns in order to achieve pattern recognition and in identifying primitives that describe patterns [9]. Such an analysis is performed by using different schemes such as neural networks, tree grammars, transformations, and other well known methods [10]. Syntactic PR offers complex patterns recognition and is of benefit if there is no suitable statistical method available. However, grammars and recognisers (recognition) are complex to implement especially with noise [11].

Associative memory (AM) is adopted by Haihong (Zhang) et al. [12] for high performance pattern recognition. Hopfield [13] implements AM by adapting synaptic weights. Hopfield is a tightly coupled network where nodes are highly interconnected and inter-dependent. Consequently, implementing this approach in WSN will result in increasing the frequency, scale, and complexity of message passing within WSN. Moreover, pre-determination of the synaptic relationship between nodes will degrade its suitability for real time applications. Morphological associative memories (MAM) [14], implement one shot learning and provide noisy pattern recognition by generating maximum and minimum matrices. However, there are two major drawbacks in using this scheme for WSN. Firstly, MAM is a tightly coupled network. Secondly, the learning cycle duration in MAM cannot be easily predicted owing to the scheme's reliance on global network communication for its computations [15]. Convolutional neural networks [16],[17], use multiple layers to reduce their complexity and inter-neuron connectivity. However, the scheme depends on learning rules and requires iterative training. Additionally, the connectivity between neurons is reduced by making part of the connections active, which means that a large number of connections are required whilst only a small number of these connections are actually used. Cortes and Vapnik [18] make use of Support-Vector Machines (SVM) for pattern recognition. However, SVM requires separation between each group of its pattern representation vectors. Doing so will spread the communications and computations across the entire network. Furthermore, SVM needs a kernel function, which would result in a densely connected network.

Graph Neuron (GN) [19],[20] is a scheme that creates AM in a fully parallel-distributed manner over fine grained WSN. GN nodes only communicate adjacently and in a loosely coupled fashion. Hence, GN offers light weight one shot learning capabilities in a decentralised manner. These characteristics make GN a very suitable approach for real time pattern recognition in WSN. In order to perform pattern recognition, each node initialises a memory structure called the *bias array* where in it stores the incoming pattern as sets of *p(value, position)* pairs. Each input pattern is automatically synthesised into its components by the GN array. The GN nodes corresponding to the respective *p(value, position)* pairs are activated by the input pattern. Each activated node exchanges its *value* and *position* with its neighbouring nodes (i.e. previous and next). In the memorisation process, a node will store the combinations of

its own value and its neighbours' values. For the recall process, it will look up the bias array for a matching combination. The node raises a recall, a yes vote, if the combination is found in its bias array. If all neurons vote yes then the input pattern will be recalled by the network. Figure 1 illustrates the architecture and communications between GN nodes in a four-position GN array with two possible values A and B; storing pattern ABBA.

The recognition accuracy of GN is affected by the limited perspective of each neuron, as each node only knows about its immediate neighbours. This leads to the crosstalk problem. For example, if a GN network with a pattern size of five had memorised patterns *abcdf* and *fbcde*, the network will falsely recall the pattern *abcde*. Hierarchal Graph Neuron (HGN) [15] fixes the crosstalk problem by using a pyramidal framework for obtaining a higher perspective of the incoming pattern. Distributed HGN (DHGN) [21] reduces the learning cycle duration and the complexity of HGN. However, the size of DHGN networks can still increase quite substantially for larger and more complex patterns.

The proposed CMPR in this paper is a light weight pattern recognition scheme, which is better suited for WSN. This scheme will address some of existing issues of other schemes by substantially reducing the number of nodes within the WSN while maintaining the pattern recognition accuracy and one shot learning capability of the GN. The next section will discuss the proposed scheme further.

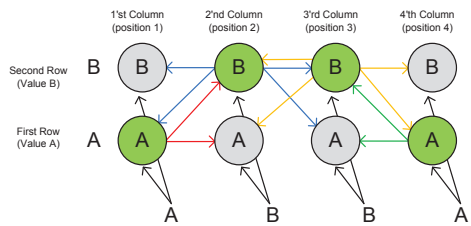

Fig. 1. A Simple four node GN array responds and stores the incoming pattern ABBA.

3. Cellular Microscopic Pattern Recogniser Scheme

The CMPR network is composed of multiple tracks, where each track contains an odd number of neuron positions. Each neuron position consists of a number of nodes that equals the number of possible values as shown in Figure 2. In this example, a pattern of size 9 and two possible values (i.e. 0 and 1) can be adopted by a network structure that contains 9 neuron positions and two nodes in each position to represent the two possible values. The inner most track should have only one neuron position called the *core position* or the *core node,* which is considered to have the highest authority over the entire network. The tracks structure can be formed by using the first position in the core of the network, and then using the following odd number of positions in the next outer track. In the case of the example shown in Figure 2, the first position includes two nodes and it is used as the core position of the network (position 1), then, three positions are used for the outer track to the core of the network (positions: 2, 3, and 4) and finally, five neuron positions are used for the outer most track. Each neuron position receives part of a pattern, exchanges information with adjacent nodes, and reports to its corresponding inner nodes. This means that if a pattern 110001001 is presented to the network, the positions 1,2,3,4,5,6,7,8, and 9 will receive the values 1,1,0,0,0,1,0,0,1 respectively. Every node, in each position, will respond according to its pre-set value. If the node is assigned the value of 0 and the received pattern is 0 then, it will be activated and start communicating with neighbouring nodes. Otherwise, the node becomes inactive. When a node is activated, it exchanges information with previous and next active nodes in the same track, determines the incoming pattern, and report to the inner track nodes. For example, the active node in position 5, in Figure 2, will communicate with active nodes in positions 6, and 9, and then report the calculation outcomes to the active nodes in positions 2, 3, and 4.

Determining the incoming pattern is based on the combinations of received pattern elements. If the pattern is to be memorised, the node explicitly stores these combinations and associate these combinations with an index number in the memory. Each index number denotes a unique pattern and the numbering is incremental process. For example, if the combination 001 is to be memorised first, the node will store this combination and associate it with index number 1 in its memory. With a new combination 011 is to be memorised next, the node will assign it a new index

number of 2. If the incoming pattern combination is already associated with an index number, the node doesn't store the combinations again and the node will use the existing associated index number for reporting purposes. The index number, in the memorisation process, may be determined according to the following relationship:

$$x_i = \begin{cases} I_i + 1, & if\ no\ match\ found \\ I_{li}, & if\ a\ match\ is\ found \end{cases} \quad (1)$$

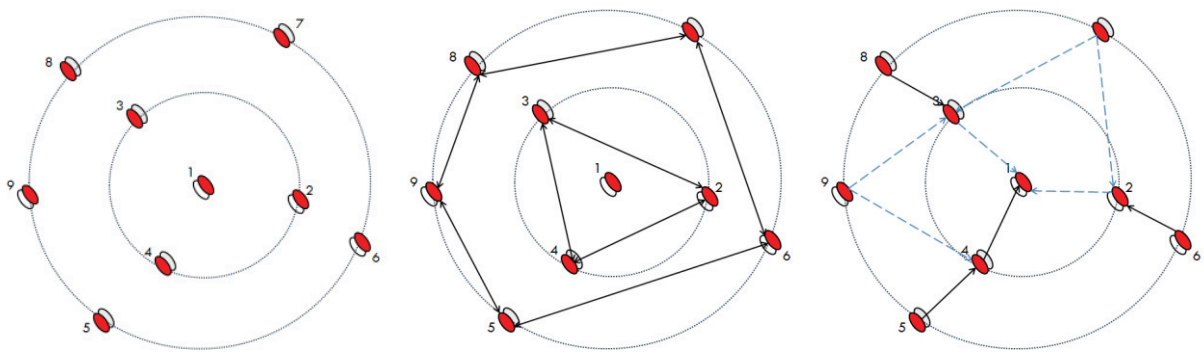

Fig. 2. The cell GN architecture for pattern size=9 and two possible values. (a) Deployed sensor nodes in the field of interest for a pattern of size 9. Red (shaded) nodes represent the activation process in each position corresponding to the pattern 110001001; (b) Simultaneous communications between adjacent nodes; (c) Reporting to inner tracks. Two GN positions per track are relieved from directly reporting to inner GNs and their indirect reporting paths are shown as light blue (dotted) arrows. The solid black arrows show how GNs report progressively to the inner track GNs.

Where x_i is the output index number of node i, $I_i$is the number of stored indices in node i, and $I_{li}$is the index number associated to a located pattern combinations in the memory of node i. Alternatively, when a pattern is to be recalled, the node searches its memory for a match combination and replies by sending the associated index number. If no match is found, the node sends index number 0 which means that the incoming pattern combinations have not been encountered before. The index number determination in the recall process may be described by the following relationship:

$$x_i = \begin{cases} 0, & if\ no\ match\ found \\ I_{li}, & if\ a\ match\ is\ found \end{cases} \quad (2)$$

Since each position in the CMPR architecture is eligible to receive part of the pattern, no higher-up GN nodes are required (in contrast with HGN and DHGN). On the other hand, the wider perspective over the pattern is attained by reporting to inner nodes. As a result, the size of the network does not increase inordinately as the pattern size increases. The CMPR network size is computed by a simple relationship:

$$N(a)=v.a \quad (3)$$

Where $N(a)$ is the number of required nodes to accommodate a pattern of size a and possible values of v. Hence a CMPR network is equal to the size of the single layered GN (which is susceptible to crosstalk), while providing the accuracy of the multi-layered HGN. CMPR requires odd number of nodes in each track. The number of required tracks to implement the CMPR network architecture can be obtained using the ceiling of the square root of the pattern size according to the following equation:

$$N_{trks}=\lceil\sqrt{a}\rceil \quad (4)$$

Where N_{trks} is the number of required tracks in a CMPR network for a pattern of size a. Hence, padding is necessary to accommodate some pattern sizes. The number of padding positions may be calculated according to the following formula:

$$N_{padding}=(N_{trks})^2-a \quad (5)$$

Where $N_{padding}$ denotes the number of padding positions. As a result, the size of the CMPR network including padding nodes may be computed according to the following formula:

$$N(a)=v.(\lceil\sqrt{a}\rceil)^2 \quad (6)$$

To illustrate above equations, let the pattern size be 14 and the possible values for each pattern position are either 0 or 1. From (4), we can calculate the number of required tracks by taking the ceiling of the square root value of 14 (3.74). The result from (4) will indicate to the need for 4 tracks to create the network structure to accommodate this pattern size. The number of padding positions according to (5) in this case will be $(4)^2$-14= 2 positions. Finally, from (6), the total number of needed nodes to accommodate that pattern will be $2x(4)^2$ = 32 nodes, where each node in each position is assigned a specific value (either 0 or 1). Figure 3 shows the number of required nodes according to pattern size with two possible values. It may be seen from the figure that the CMPR network size grows steadily with the increase in pattern size (whereas, the HGN network size grows exponentially).

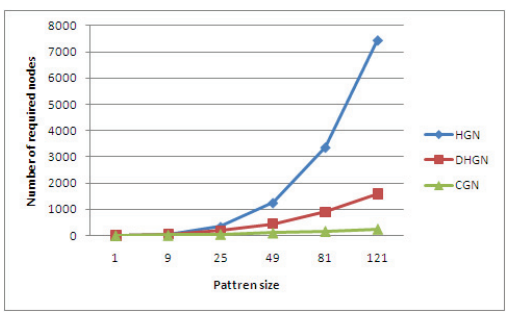

Fig. 3. Required number of nodes depending on the pattern size with two possible values for HGN, DHGN, and CMPR

3.1. Communication scheme

The CMPR scheme employs an input/output program called Stimulator & Interpreter module (S&I) for sending the patterns and obtaining the results. The S&I operations can be performed by a single entity in the network (i.e. *base station*). Communication in CMPR goes through the following stages;

1. The S&I sends the pattern with a command to all GN positions in the network, where each GN position is responsible for receiving a small part (pair) of the incoming pattern. The command would be either to memorise or to recall the pattern.
2. Activated GNs in each position report to adjacent nodes in the same track by sending messages that contain their positions and values.
3. Based on the received messages from adjacent nodes and the S&I, active GNs inspect their bias array entries, determine the index number, and report to the inner track according to their positions. If an active GN cannot find the index number in the recall operation, it sends 0 as the result of inspecting the pattern in the bias array. Two nodes in each track will not directly link to inner GNs and will be called *neglected nodes*. Each of the other active nodes will report its index number to its corresponding inner node and two neighbour nodes. The

neglected nodes will only report to their penultimate GNs. This reporting protocol eliminates the crosstalk problem.

4. The core GN replies to the S&I with the index number. In memorisation stage, the S&I will save the index number to the corresponding input pattern in its database. In recall stage, the returned index number will denote the detected pattern.

Figure 2(c) shows how the GNs report to inner tracks. The black arrows (solid lines) indicate non-neglected node direct reports while light blue arrows (dotted lines) indicate indirect reports from neglected GNs. It is to note that GNs in the outer most track will receive three reports: the S&I (current), the predecessor node, and the successor node reports. GNs in the middle tracks will receive six reports, and the core node will receive four reports. Thus, the maximum bias array entry length will be equal to six.

3.2. Complexity of the CMPR scheme

As the scheme allows individual GN nodes to function in parallel by exchanging information with adjacent and inner nodes simultaneously, the learning cycle duration may be estimated using the number of tracks, the number of entries in the bias array of a GN, and the time required to search an entry in the bias array. This section describes the calculation of the CMPR learning duration by using terms described in Table 1. The time needed for all nodes to exchange information in the same track, where each activated node sends two messages to its adjacent activated nodes, may be calculated according to the following equation:

$$T_1 = 2.(N_{trks}-1).(T_{send}+T_{overhead}) \quad (7)$$

Table 1. Description of the terms used for complexity estimation

Symbol	Name	Description
S	Size	Network size in terms of positions including padding positions, where S≥ 4.
T1	Exchange time	Time required for all nodes to exchange information in the same track.
T2	Total search time	Time required for all nodes to search their bias arrays.
T3	Reporting time	Time required for all nodes to report to corresponding inner nodes.
Ttotal	Total time	The total time of the learning/recall cycle.
Tsend	Sending time	Time to send a message from one node to another.
Toverhead	Overhead time	Small overhead time per node. (i.e. hardware latency)
Tbias	Bias time	Time for a node to memorise or recall a pattern within its bias array.
Nentries	Number of entries	Maximum number of entries within the bias array.
Ntrks	Number of tracks	Number of tracks that form the CMPR network.

As the core node has no adjacent nodes, the core track has been excluded from T1 in equation 7. The computation time required to find a match in the bias array is dependent on the number of entries within the bias array. As all nodes in each track perform their computations simultaneously, the computation time for each track

will be equal to the highest computation time for any one node. The total time will be dependent on the number of tracks rather than the number of nodes. Consequently, the total time needed to search bias arrays in the network may be estimated as follows.

$$T_2 = T_{bias}.N_{entries}.N_{trks} \quad (8)$$

The total reporting time needed in the CMPR learning cycle is dependent on the number of reporting messages that nodes send to their adjacent and associated inner nodes. The maximum number of messages for a node to report to an inner track node is equal to 3. This means that if a node is assigned an inner node, it will send its reports to the assigned inner node and its adjacent nodes. This applies for all tracks except the inner most track and its outer track. In the outer track to the core node, there is only one message per node used for reporting as the core node has no adjacent nodes. In addition, all reporting messages are sent simultaneously in each track (making the reporting time dependent on the number of tracks rather than number of nodes). The total reporting time may be calculated by the following equation:

$$T_3 = (3.(N_{trks}-2)+1).(T_{send}+T_{overhead}) \quad (9)$$

The total learning and recall cycle is the summation of the three times T1,T2, and T3. This allows to estimate the total time according to the following formula:

$$T_{total} = (5.\ N_{trks}-7).\ (T_{send}+T_{overhead})+\ T_{bias}.N_{entries}.N_{trks} \quad (10)$$

From (4), it is possible to obtain the total time in terms of the pattern size according to the following equation:

$$T_{total} = (5.\ \sqrt{S}\ -7).\ (T_{send}+T_{overhead})+\ T_{bias}.N_{entries}.\ \sqrt{S} \quad (11)$$

The above relations are valid for on any CMPR network configuration with $S \geq 4$ i.e. pattern size greater than unity. From equation (11), the total time depends on the square root of the number of positions that represent the pattern size rather than on the pattern size itself. Additionally, the learning cycle duration is independent of the number of GNs in each position. As a result, the scheme will scale-up to support large pattern sizes with moderate increase in the total time.

3.3. CMPR as a pattern recogniser

The Pattern recognition application for the CMPR scheme consists of two main elements: The S&I and the GN arrays. The S&I sends the pattern and the command to the GN arrays and receives the result from the core node. If the core node cannot provide an index number, the S&I will go to the outer track to obtain the higher index occurrence percentage and consider it as the detected pattern. If one or more GNs could not come up with an index number in any track, the S&I will go to the next outer track while keeping records of voted GNs. This continues until the S&I contacts a track where all GNs have voted or it reaches the outer most track. If a GN sends an index number, it means that it votes for that index. Its corresponding outer GNs will also be counted as voters for the same index, on the basis that the GN in an inner track has a higher perspective than a GN in the outer one.

4. Tests and Results

We have run three test series to examine the CMPR. In the first test, we checked the scheme against crosstalk. In the second and third tests, we performed bitmap image recognition to establish scheme's recognition accuracy.

In the first test, we tested the CMPR against crosstalk. The crosstalk phenomenon occurs if two or more stored patterns result in the recognition of a new pattern which has not been memorised before. In this test, we created a

CMPR network as an identifier to analyse binary patterns of size=9. Choosing the CMPR as identifier network means that the S&I gets the result only from the core position which assures that patterns are recalled based upon sub-pattern combinations and fault-tolerance features have been avoided. The number of all possible patterns was 512. We then stored 100 random patterns in the CMPR network. Finally, all the 512 possible patterns were sent by the S&I to be recalled by the CMPR. This test was repeated one hundred times and there were no false matches found, i.e. the memorised patterns were the only patterns recalled by the CMPR. By performing this test one hundred times, we increased the probability of memorising two deferent patterns that produce sub-patterns of patterns that have not been memorised among the 100 randomly selected patterns in each iteration. This demonstrates that the scheme is not susceptible to the crosstalk problem.

In the second series test, we represented letters “A”, ”I”, “J”, “S”, “X”, and “Z” as bitmap images of size 7x5, and created a CMPR of size 36 with one padding position to memorise these images. These letters are not very close in terms of pattern characteristics. Each image was then randomly distorted to varying degrees ranging from 1bit (2.78%) to 15bits (41.67%). Figure 4 shows samples of recall results for distorted stored patterns. Each result represents the percentage of GNs that responded to the resultant index in the network. This test confirms the CMPR scheme is capable of detecting distorted patterns. The results also show the ability of the CMPR network to detect patterns of characters “A”, ”I”, “J”, and “S” with a high level of distortion - 13bit distortion level (36.11%). The accuracy of the CMPR network’s recalls is shown in Figure 5. Figure 6 presents a comparison between the average recall accuracy of the CMPR and the HGN. The figure shows that the CMPR network has comparable average recall accuracy despite using far fewer processing nodes than the HGN. For instance The HGN requires 648 nodes to represent the 35 length binary patterns in comparison with 72 nodes required by the CMPR; reduction of 88.89% in the network size.

In the third test series, we modelled letters “A”, “E”, “F”, “K”, and “T” as binary bitmap image patterns of size 7x5 with one padded position. These letters share edge characteristics in the binary representation. We then applied three levels of distortion to each stored pattern: Low level (<5%), Medium level (<11%), and High level distortions (>11%). Figure 6 shows the average accuracy ratio of the CMPR for 100 randomly distorted patterns compared with those of the DHGN. It may also be seen that the CMPR provides comparable recognition accuracy to the DHGN with a cut down of 42.86% in the network size.

Memorized Patterns	00100 01010 10001 11111 10001 10001 10001 0	11111 00100 00100 00100 00100 00100 11111 0	11111 00001 00001 10001 10001 10001 01110 0	01111 10000 10000 01110 00001 00001 01111 0	10001 10001 01010 00100 01010 10001 10001 0	11111 00001 00010 00100 01000 10000 11111 0
Distorted (5 bits) (13.89%)	00110 01010 00000 11111 10000 10011 10001 0	11110 00100 00100 00100 00110 00110 00111 0	01111 10001 00001 11001 11001 00001 01110 0	01111 00010 10101 01110 00001 00001 01011 0	10011 10000 00010 00100 00110 10001 10001 0	11011 00001 11010 00101 01100 10000 11111 0
Result	20 A	40 I	25.71 J	11.43 S	11.43 X	8.57 Z
Distorted (9 bits) (25%)	00110 00010 10000 10110 10000 11001 00101 0	11111 01111 00100 11101 01100 00100 01011 0	11101 11001 00011 10001 11000 10110 01110 0	10111 10001 11001 01110 00001 01001 10110 0	00000 11001 01010 01100 00100 10011 11001 0	11111 01001 01010 01111 00001 10100 11011 0
Result	11.43 S	20 I	20 J	14.29 S	11.43 X	11.43 Z
Distorted (13 bits) (36.11%)	00100 10010 00101 10001 01001 10101 00100 0	01101 01100 00000 01000 01110 11100 01001 0	01011 10111 10101 00101 11001 10000 01011 0	01010 01000 10101 11110 01001 00110 01100 0	00101 10100 01011 00111 00010 10111 01001 1	10011 00010 01111 00100 11100 11101 11111 0
Result	11.43 A	5.71 I	2.86 J	17.14 S	8.57 I	5.71 S
Distorted (15 bits) (41.67%)	00100 01111 01011 11010 10110 10110 11001 1	01111 00001 11111 01010 10100 11001 11111 0	11110 11101 01111 00101 11001 10100 01101 1	11010 10010 00000 00111 11011 01100 01100 0	01000 10110 00100 10100 11010 10001 00001 1	00101 10001 00010 11011 00100 11001 10111 1
Result	11.43 A	11.43 S	5.71 I	5.71 I	2.86 I	8.57 J

Fig. 4. Patterns recalled with distortions ranging from 2.78% to 41.67% by the CMPR.

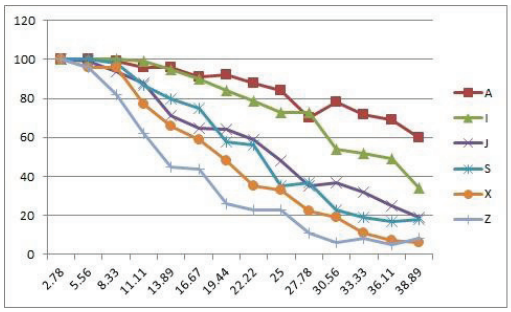

Fig. 5. Accuracy recall percentage for the CMPR using 100 randomly distorted patterns per memorised pattern.

In the third test series, we modelled letters “A”, “E”, “F”, “K”, and “T” as binary bitmap image patterns of size 7x5 with one padded position. These letters share edge characteristics in the binary representation. We then applied three levels of distortion to each stored pattern: Low level (<5%), Medium level (<11%), and High level distortions (>11%). Figure 6 shows the average accuracy ratio of the CMPR for 100 randomly distorted patterns compared with those of the DHGN. It may also be seen that the CMPR provides comparable recognition accuracy to the DHGN with a cut down of 42.86% in the network size.

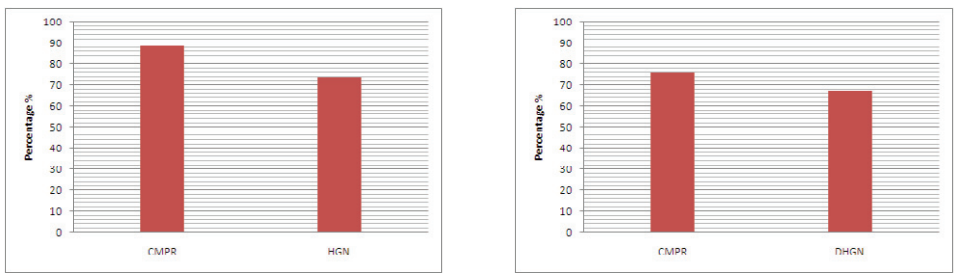

Fig. 6. (a) Test 2 average recall accuracy of CMPR compared to HGN using 100 randomly distorted images for each distortion level per pattern, (b) Test 3 average recall accuracy of CMPR compared to DHGN using 100 randomly distorted images for each distortion level per pattern.

The tests results show that the proposed scheme is capable of performing pattern recognition in a single learning cycle by distributing the computations among network elements. Additionally, it maintains comparable recognition accuracy levels with substantial reduction in the number of required nodes compared to other schemes. The scheme is also capable of dealing with noisy patterns which suits the nature of WSN.

5. Conclusion

Detection of complex phenomena, occurring over large spaces, becomes a distinct possibility with advents of light-weight distributed pattern recognition schemes such as CMPR for WSN. Patterns of macroscopic event for WSN are not available at present. Hence samples of bitmap images that represent visibly associative patterns have been used for obtaining tests results. The proposed scheme is however capable of handling generic patterns. For future research, the CMPR network deployment in real field will be studied, and the scheme will be tested with live data.

References

1. S. Kim, *et al.*, "Health monitoring of civil infrastructures using wireless sensor networks," 2007, pp. 254-263.
2. T. Kijewski-Correa, *et al.*, "Wireless sensor networks for structural health monitoring: a multi-scale approach," 2006.
3. D. Lymberopoulos, *et al.*, "Extracting spatiotemporal human activity patterns in assisted living using a home sensor network," presented at the Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments, Athens, Greece, 2008.

4. A. Iyer, *et al.*, "A Taxonomy-based Approach to Design of Large-scale Sensor Networks," in *Wireless Sensor Networks and Applications*, Y. Li, *et al.*, Eds., ed: Springer US, 2008, pp. 3-33.
5. D. Chen and P. Varshney, "QoS support in wireless sensor networks: A survey," 2004, pp. 227-233.
6. Y. Kim, *et al.*, "Design of a fence surveillance system based on wireless sensor networks," 2008, pp. 1-7.
7. T. Bokareva, *et al.*, "Wireless sensor networks for battlefield surveillance," 2006.
8. R. O. Duda, *et al.*, *Pattern classification*, 2nd ed. New York ; Chichester: Wiley, 2001.
9. R. Rengaswamy and V. Venkatasubramanian, "A syntactic pattern-recognition approach for process monitoring and fault diagnosis," *Engineering Applications of Artificial Intelligence,* vol. 8, pp. 35-51, 1995.
10. F. King-Sun and B. K. Bhargava, "Tree Systems for Syntactic Pattern Recognition," *Computers, IEEE Transactions on,* vol. C-22, pp. 1087-1099, 1973.
11. R. J. Hamilton, *et al.*, "Syntactic techniques for pattern recognition on sampled data signals," *Computers and Digital Techniques, IEE Proceedings E,* vol. 139, pp. 156-164, 1992.
12. Z. Haihong, *et al.*, "Gabor wavelet associative memory for face recognition," *Neural Networks, IEEE Transactions on,* vol. 16, pp. 275-278, 2005.
13. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences of the United States of America,* vol. 79, p. 2554, 1982.
14. G. X. Ritter, *et al.*, "Morphological associative memories," *Neural Networks, IEEE Transactions on,* vol. 9, pp. 281-293, 1998.
15. B. B. Nasution and A. I. Khan, "A Hierarchical Graph Neuron Scheme for Real-Time Pattern Recognition," *Neural Networks, IEEE Transactions on,* vol. 19, pp. 212-229, 2008.
16. Y. L. Cun, *et al.*, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems 2*, ed: Morgan Kaufmann Publishers Inc., 1990, pp. 396-404.
17. Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks,* pp. 255–258, 1995.
18. C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning,* vol. 20, pp. 273-297, 1995.
19. A. Khan, "A peer-to-peer associative memory network for intelligent information systems," 2002.
20. A. I. Khan, *et al.*, "A parallel distributed application of the wireless sensor network," in *High Performance Computing and Grid in Asia Pacific Region, 2004. Proceedings. Seventh International Conference on*, 2004, pp. 81-88.
21. A. I. Khan and A. H. M. Amin, "One shot associative memory method for distorted pattern recognition," presented at the Proceedings of the 20th Australian joint conference on Advances in artificial intelligence, Gold Coast, Australia, 2007.